1 Linked Lists

Questions

1.1 What is a linked list? Why do we consider it a naturally recursive structure?

1.2 Draw a box and pointer diagram for the following:

\[
\text{Link('c', Link(6, Link(1, Link('a'))), Link('s'))}
\]

1.3 The Link class can represent lists with cycles. That is, a list may contain itself as a sublist. Implement has_cycle that returns whether its argument, a Link instance, contains a cycle. There are two ways to do this: iteratively with two pointers, or keeping track of Link objects we’ve seen already. Try to come up with both!

```python
def has_cycle(link):
    """
    >>> s = Link(1, Link(2, Link(3)))
    >>> s.rest.rest.rest = s
    >>> has_cycle(s)
    True
    """
```

1.4 Fill in the following function, which checks to see if sub_link, a particular sequence of items in one linked list, can be found in another linked list (the items have to be in order, but not necessarily consecutive).

```python
def seq_in_link(link, sub_link):
    """
    >>> lnk1 = Link(1, Link(2, Link(3, Link(4)))))
    >>> lnk2 = Link(1, Link(3))
    >>> lnk3 = Link(4, Link(3, Link(2, Link(1)))))
    >>> seq_in_link(lnk1, lnk2)
    True
    >>> seq_in_link(lnk1, lnk3)
```

False
2 OOP

Questions

2.1 What is the relationship between a class and an ADT?

2.2 What is the definition of a Class? What is the definition of an Instance?

2.3 What is a Class Attribute? What is an Instance Attribute?

2.4 What Would Python Display?

```python
class Foo:
    x = 'bam'
    def __init__(self, x):
        self.x = x
    def baz(self):
        return self.x

class Bar(Foo):
    x = 'boom'
    def __init__(self, x):
        Foo.__init__(self, 'er' + x)
    def baz(self):
        return Bar.x + Foo.baz(self)

foo = Foo('boo')

Foo.x

foo.x

foo.baz()

Foo.baz()

Foo.baz(foo)

bar = Bar('ang')
Bar.x

bar.x

bar.baz()
```

Note: This worksheet is a problem bank—most TAs will not cover all the problems in discussion section.
2.5 What Would Python Display?

```python
class Student:
    def __init__(self, subjects):
        self.current_units = 16
        self.subjects_to_take = subjects
        self.subjects_learned = {}
        self.partner = None

    def learn(self, subject, units):
        print(f'I just learned about {subject}')
        self.subjects_learned[subject] = units
        self.current_units -= units

    def make_friends(self):
        if len(self.subjects_to_take) > 3:
            print('Whoa! I need more help!')
            self.partner = Student(self.subjects_to_take[1:])
        else:
            print('I’m on my own now!')
            self.partner = None

    def take_course(self):
        course = self.subjects_to_take.pop()
        self.learn(course, 4)
        if self.partner:
            print('I need to switch this up!')
            self.partner = self.partner.partner
        if not self.partner:
            print('I have failed to make a friend :(')

tim = Student(['Chem1A', 'Bio1B', 'CS61A', 'CS70', 'CogSci1'])
tim.make_friends()

print(tim.subjects_to_take)
tim.partner.make_friends()
tim.take_course()
tim.partner.take_course()
tim.take_course()
tim.make_friends()
```

Note: This worksheet is a problem bank—most TAs will not cover all the problems in discussion section.
Fill in the implementation for the Cat and Kitten classes. When a cat meows, it should say "Meow, (name) is hungry" if it is hungry, and "Meow, my name is (name)" if not. Kittens do the same thing as cats, except they say "I'm baby" instead of "meow", and they say "I want mama (parents name)" after every call to meow().

```python
>>> cat = Cat('Tuna')
>>> kitten = Kitten('Fish', cat)
>>> cat.meow()
meow, Tuna is hungry
>>> kitten.meow()
i'm baby, Fish is hungry
 I want mama Tuna
>>> cat.eat()
meow
>>> cat.meow()
meow, my name is Tuna
>>> kitten.eat()
i'm baby
>>> kitten.meow()
meow, my name is Fish
 I want mama Tuna

class Cat():
    noise = 'meow'
    def __init__(self, name):
        self.hungry = True

    def meow(self):
        print(self.noise)
        self.hungry = False

class Kitten(Cat):
```

Note: This worksheet is a problem bank—most TAs will not cover all the problems in discussion section.